Feature Extraction from Telematics Car Driving Heatmaps

Guangyuan Gao*† Mario V. Wüthrich‡

November 13, 2017

Abstract

Insurance companies have started to collect high-frequency GPS car driving data to analyze the driving styles of their policyholders. In previous work, we have introduced speed and acceleration heatmaps. These heatmaps were categorized with the K-means algorithm to differentiate varying driving styles. In many situations it is useful to have low-dimensional continuous representations instead of unordered categories. In the present work we use singular value decomposition and bottleneck neural networks for principal component analysis. We show that a two-dimensional representation is sufficient to re-construct the heatmaps with high accuracy (measured by Kullback-Leibler divergences).

Keywords. Telematics car driving data, driving styles, unsupervised learning, pattern recognition, image recognition, bottleneck neural network, singular value decomposition, principal component analysis, Kullback-Leibler divergence.

1 Introduction

In many fields of business more and more detailed data becomes available. General insurance companies have started to collect telematics car driving data which comprises high-frequency GPS location data. This data allows companies to obtain driver-individual information, for instance, about speed and force of acceleration. This telematics car driving data has started to stimulate new research in the field of car insurance pricing. One branch of research considers so-called usage-based insurance (UBI) and pay-as-you-drive (PAYD) covers, see for instance Ayuso et al. [1]. This analysis mainly focuses on the question of the right exposure measure for car insurance; this has also been analyzed in related work by Verbelen et al. [6]. This kind of analysis, however, rather focuses on driving habits but less on driving styles. Recent work that analyzes driving styles is the research of Weidner et al. [7, 8]. These authors analyze the speed and acceleration pattern of individual car drivers using Fourier analysis decomposition. In another approach we have been analyzing so-called v-a heatmaps which graphically illustrate

*Center for Applied Statistics and School of Statistics, Renmin University of China

†Financially supported by the Social Science Fund of China (Grant No. 16ZDA052) and MOE National Key Research Bases for Humanities and Social Sciences (Grant No. 16JJD910001)

‡ETH Zurich, RiskLab, Department of Mathematics, 8092 Zurich, Switzerland

Electronic copy available at: https://ssrn.com/abstract=3070069

driving styles using speed and acceleration patterns, see Wüthrich [9]. Classification of individual drivers was achieved by using the K-means algorithm.

The K-means algorithm has the disadvantage that a good choice of the number of classes K is often difficult, and that a too large choice may imply over-parametrization in the subsequent car insurance pricing regression framework (caused by too many categorical classes). For these reasons we explore (low-dimensional) continuous representations of the v-a heatmaps in this work. Continuous features have the advantage that they may provide better properties in regression models such as generalized linear models or generalized additive models. The most straightforward method of dimension reduction is principal component analysis (PCA) using singular value decomposition (SVD). This is a linear method that explores the feature space for the direction of the biggest variance. We provide this method in Section 3.1, below. A second approach of dimension reduction that we consider is a non-linear approach that is based on a so-called bottleneck neural network. This technique has been introduced by Kramer [2] and it has recently been used by Hainaut [3] in a mortality modeling context. A bottleneck neural network is a feed-forward neural network with one hidden layer being very low-dimensional. Since all signals need to traverse this low-dimensional layer (bottleneck) we receive a low-dimensional representation of the (high-dimensional) input signals at the bottleneck, this is described in Section 3.2. In Section 4 we compare the methods on the explicit example studied in Wüthrich [9]. In the next section we describe this data.

2 Description of the data

The data used is exactly the same as in Wüthrich [9]. It comprises GPS location data second by second of $n = 1753$ individual car drivers. For each of these car drivers we have recorded 200 individual trips (which have normalized coordinates for confidentiality reasons). These 200 individual trips are illustrated for three different car drivers in Figure 2 in Wüthrich [9]. Out of this data we can calculate driving habits such as total distances, average speeds, etc. To analyze driving styles we construct so-called v-a heatmaps. For this purpose we consider the following speed buckets (in km/h):

$[0]$	car stands still (and also does not accelerate),
$(0, 5]$	acceleration or braking phase (from/to speed 0),
$(5, 20]$	low speeds,
$(20, 50]$	urban area speeds,
$(50, 80]$	rural area speeds,
$(80, 130]$	highway speeds (we truncate speeds above 130 km/h).

For each of these speed buckets we calculate the amount of time spend by individual drivers at a certain acceleration or braking level (we denote speed by $v \geq 0$ and acceleration and braking by $a \in \mathbb{R}$). That is, we calculate for each second $t \geq 1$ and each trip the average speed

$$v_t = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}/(t - (t-1)),$$

Electronic copy available at: https://ssrn.com/abstract=3070069

where (x_t, y_t) is the GPS location at time $t \geq 0$ of that trip (transformed to meters). This average speed v_t has unit m/s; for the illustrations we typically transform it to km/h. These average speeds allow us to calculate an average acceleration and braking, for $t \geq 2$, defined by

$$a_t = (v_t - v_{t-1})/(t - (t-1)).$$

For illustrative purposes we consider speed bucket $(5, 20]$ km/h. For each driver $i = 1, \ldots, n$ we then construct the corresponding v-a heatmap by considering all observations (v_t, a_t) with speed v_t being in speed bucket $(5, 20]$ km/h. If we normalize these observations by the number of observations that fall into that speed bucket we get a two-dimensional (discrete) distribution on the rectangle $R = (5, 20] \times [-2, 2]$. For convenience we truncated acceleration and braking a_t at ± 2 m/s^2; note that the inertia of cars does not allow for arbitrary acceleration and braking, see also Weidner et al. [8]. In Figure 1 we illustrate these v-a heatmaps for the four car drivers

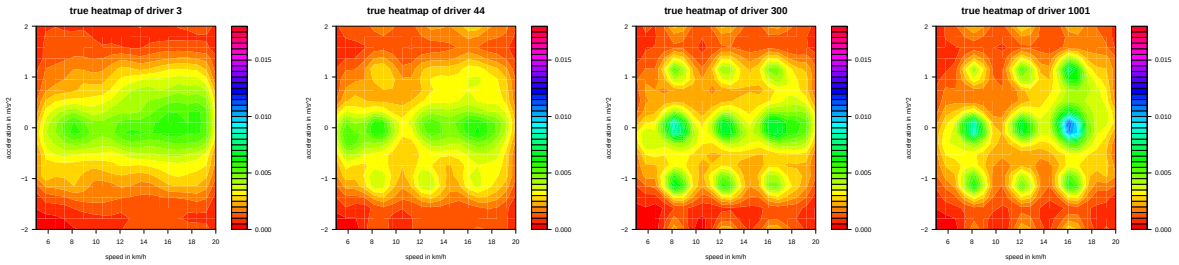

Figure 1: v-a heatmaps in speed bucket $(5, 20]$ km/h of car drivers $i = 3, 44, 300, 1001$.

$i = 3, 44, 300, 1001$. These plots show substantial differences in driving styles. The main difficulty now is to extract these differences as covariate information for car insurance pricing, for instance, in a regression analysis. In Wüthrich [9] we have classified these v-a heatmaps using the K-means algorithm. The resulting classes for $K = 4$ are presented in Figure 2 (these are exactly

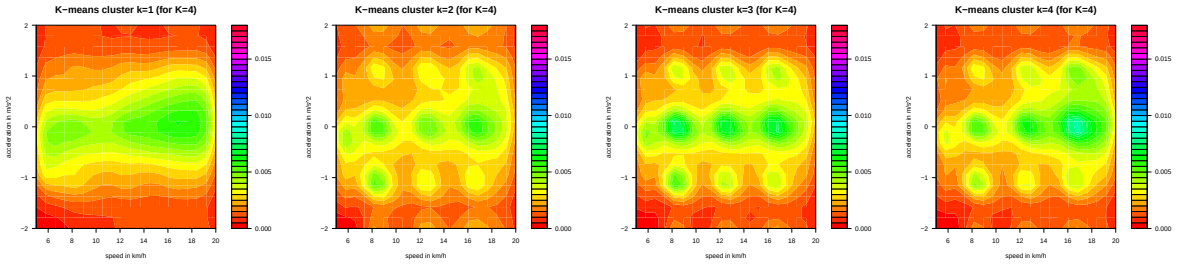

Figure 2: K-means classification of the v-a heatmaps in speed bucket $(5, 20]$ km/h for $K = 4$.

the classes obtained in Figure 6 of Wüthrich [9], we have only changed the color scale because this will be more convenient below). Note that driver $i = 3$ falls into class $k = 1$, driver $i = 44$ into class $k = 2$, driver $i = 300$ into class $k = 3$ and driver $i = 1001$ into class $k = 4$ (compare Figures 1 and 2). Class 1 contains 376 drivers, class 2 has 515 drivers, class 3 has 440 drivers and

class 4 has 422 drivers. In view of Figures 1 and 2 we may question this classification because the individual car drivers show some differences to the overall averages in the classes chosen. This raises the question of an appropriate number K of classes in the K-means algorithm. Another issue is that categorical features are more difficult in generalized linear model and generalized additive model applications because they lead more likely to over-parametrization. For this reason we explore continuous covariate extraction in this paper.

3 Dimension reduction using principal component analysis

The idea is to replace the categorical classes provided by the K-means algorithm by low-dimensional continuous features. The first idea is to use the SVD for a PCA of the v-a heatmaps given in Figure 1. The SVD is linear in nature, therefore we explore a neural network in a second step that leads to a non-linear PCA.

3.1 Singular value decomposition

We start by describing the SVD. First, we discretize the data in rectangle $R = (5, 20] \times [-2, 2]$ by dividing both axes into 20 equidistant intervals. This partitions R into $J = 20^2 = 400$ congruent sub-rectangles $(R_j)_{j=1:J}$. Note that J is comparably small; we do this small choice for computational efficiency. For each driver $i = 1, \ldots, n$ we denote by $x_{i,j}$ the relative amount of time spent in sub-rectangle $R_j \subset R$. This provides for all car drivers $i = 1, \ldots, n$ discrete distributions $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,J})'$ on R with

$$x_{i,j} \geq 0 \text{ for all } j = 1, \ldots, J, \text{ and } \sum_{j=1}^{J} x_{i,j} = 1,$$

see also (3.2) in Wüthrich [9]. The set $\mathcal{X}$ of all J-dimensional distributions is given by the J-unit simplex. Using our observations $\boldsymbol{x}_i \in \mathcal{X}$, $i = 1, \ldots, n$, we aim at finding a partition of the J-unit simplex $\mathcal{X}$ into homogeneous sub-classes. This is exactly what is done by the K-means algorithm illustrated above. In this section we are looking for a low-dimensional continuous representation.

We denote by $X = (\boldsymbol{x}_1', \ldots, \boldsymbol{x}_n')' \in \mathbb{R}^{n \times J}$ the $n \times J$ design matrix that contains the (discretized) v-a heatmaps of all drivers $i = 1, \ldots, n$. Denote by $X^0 \in \mathbb{R}^{n \times J}$ a normalized version of X so that all column means are zero (additionally we have standardized the column vectors to unit variance, but there may be other good choices too). Then there exists an $n \times J$ orthogonal matrix U ($U'U = \mathbb{1}_J$), a $J \times J$ orthogonal matrix V ($V'V = \mathbb{1}_J$) and a $J \times J$ diagonal matrix $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_J)$ with singular values $\lambda_1 \geq \ldots \geq \lambda_J \geq 0$ such that we have SVD

$$X^0 = U \Lambda V',$$

we refer to Section 14.5 in Hastie et al. [4]. Multiplying the latter identity from the right with V, we receive the principal components of X^0 as the columns of the matrix $X^0 V = U\Lambda = U\text{diag}(\lambda_1, \ldots, \lambda_J)$. In PCA we now analyze how many singular values $\lambda_1 \geq \ldots \geq \lambda_q$,

$1 \leq q \leq J$, and principal components (columns of $U\Lambda$) we need to approximate the true matrix X^0 sufficiently well. In Figure 4 (lhs) we provide the singular values $\lambda_1 \geq \ldots \geq \lambda_J \geq 0$ of our $n = 1753$ car driver observations given by X^0. Denote by $\Lambda_q = \text{diag}(\lambda_1, \ldots, \lambda_q, 0, \ldots, 0) \in \mathbb{R}^{J \times J}$. The best possible rank q approximation to X^0 w.r.t. the Frobenius norm is given by

$$\widehat{X}_q^0 = U\Lambda_q V',$$

and doing the back-transformation to the original column means (and variances, respectively) we obtain the best possible rank q approximation $\widehat{X}_q$ to X.

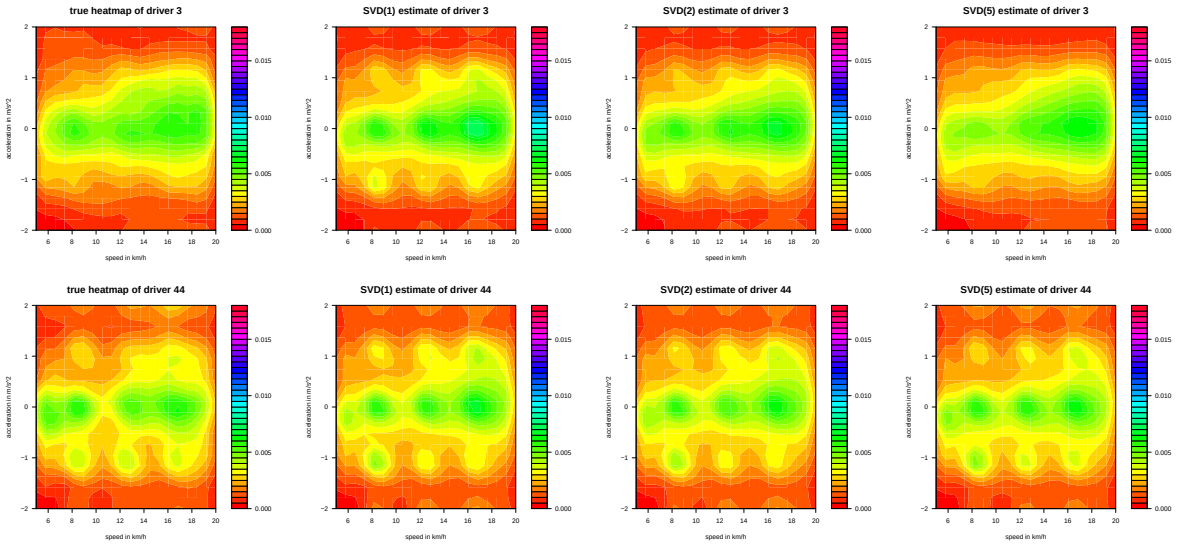

Figure 3: SVD of drivers $i = 3$ (1st row) and $i = 44$ (2nd row): (lhs) original heatmap, (other columns) heatmaps of the SVDs with ranks $q = 1, 2, 5$.

In Figure 3 we illustrate the SVD for drivers $i = 3$ (1st row) and $i = 44$ (2nd row). The 1st column provides the original heatmaps and the other columns the PCA approximations for ranks $q = 1, 2, 5$. We observe that already the first principal component provides a rather good approximation, $q = 2$ leads to an improvement and for $q = 5$ the original plot is well approximated by the one in $\widehat{X}_5$. This illustrates that the v-a heatmap can be summarized in a low-dimensional object, say of dimension $q = 2$, being the first two principal components of $U\Lambda = U\Lambda_{q=2}$. We remark that the v-a heatmaps are comparably regular and low-dimensional in which case the SVD often works well (in contrast to more complex image recognition problems). We will come back to this in Section 4.

3.2 Bottleneck feed-forward neural networks

The deficiency of SVD is that it is based on linear approximations. In this section we study two different feed-forward neural networks: a shallow one with 1 hidden layer and a deep one with 3 hidden layers. These neural networks lead to non-linear approximations of the v-a heatmaps.

We treat the two neural networks separately in the next two sections because their calibration needs to be done differently.

3.2.1 Shallow neural network approach

We start by considering a shallow neural network with q hidden neurons in the single hidden layer. These hidden neurons are given by the following hyperbolic tangent activations

$$z_l(\boldsymbol{x}) = \tanh\left(w_{l,0} + \sum_{j=1}^{J} w_{l,j} x_j\right), \qquad \text{for } l = 1, \ldots, q, \tag{3.1}$$

with feature $\boldsymbol{x} = (x_j)_{j=1:J} \in \mathcal{X}$ and weights $\boldsymbol{w} = (w_{l,j})_{l=1:q,j=0:J} \in \mathbb{R}^{q\times(J+1)}$. These hidden neurons are used for the regression equations

$$\mu_j(\boldsymbol{x}) = \mu_j(\boldsymbol{x};\theta) = \beta_0^{(j)} + \sum_{l=1}^{q} \beta_l^{(j)} z_l(\boldsymbol{x}), \qquad \text{for } j = 1, \ldots, J, \tag{3.2}$$

with network parameter $\theta = ((\beta_l^{(j)})_{j,l}, (w_{l,j})_{l,j}) \in \mathbb{R}^{J(q+1)+q(J+1)}$. An example for $q = 2$ hidden neurons (in red color) is given in Figure 4 (middle); note that for illustrative reasons we do not show all $J = 400$ neurons in the input and output layers, but only 20. These regression

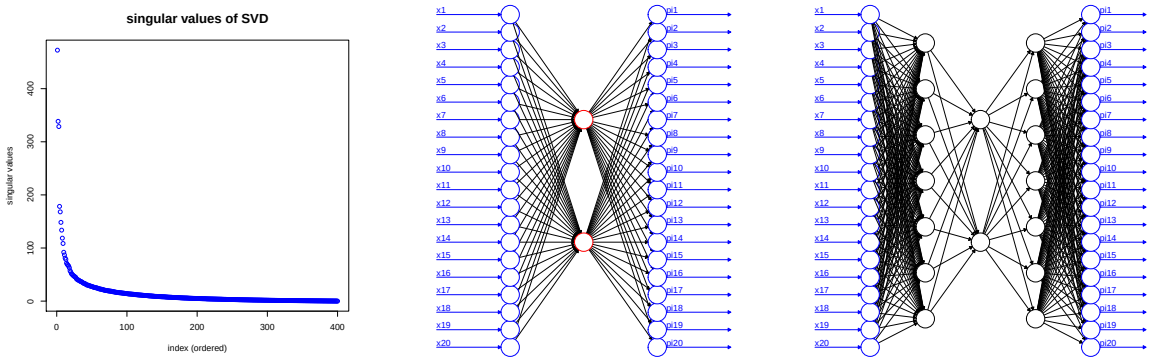

Figure 4: (lhs) singular values $\lambda_1 \geq \ldots \geq \lambda_J \geq 0$ of the $n = 1753$ car drivers; (middle) shallow neural network with $q = 2$ hidden neurons; (rhs) deep neural network with 3 hidden layers having $(p, q, p) = (7, 2, 7)$ hidden neurons; for illustrative reasons we reduce the input and output layers (blue color) from $J = 400$ neurons to 20 neurons.

equations induce the following multinomial logistic probabilities $\pi(\cdot) = (\pi_j(\cdot))_{j=1:J}$ given by

$$\pi_j(\boldsymbol{x}) = \frac{\exp\{\mu_j(\boldsymbol{x})\}}{\sum_{j'=1}^{J} \exp\{\mu_{j'}(\boldsymbol{x})\}}, \qquad \text{for all } \boldsymbol{x} \in \mathcal{X}.$$

Note that both $\boldsymbol{x} \in \mathcal{X}$ and $\pi(\boldsymbol{x}) \in \mathcal{X}$ live in the same space. Thus, the shallow neural network maps the J-unit simplex $\mathcal{X}$ onto itself. The goal now is to calibrate this neural network such that $\boldsymbol{x}_i$ and $\pi(\boldsymbol{x}_i)$ are close for all observed feature values $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$. As distance function

we consider the (average) Kullback-Leibler (KL) divergence

$$\mathcal{L}_{\mathrm{KL}}\left(\theta,(\boldsymbol{x}_i)_{i=1:n}\right)=\frac{1}{n}\sum_{i=1}^{n}d_{\mathrm{KL}}\left(\boldsymbol{x}_i\|\pi(\boldsymbol{x}_i)\right)=-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{J}x_{i,j}\log\frac{\pi_j(\boldsymbol{x}_i)}{x_{i,j}}, \tag{3.3}$$

we also refer to the appendix for more interpretation about the KL divergence. The aim is to minimize this KL divergence in the network parameter θ. A (local) minimum is found by the application of the gradient descent method (GDM), for details see Hastie et al. [4] and Wüthrich-Buser [10]. This GDM requires the calculation of the gradient of $\mathcal{L}_{\mathrm{KL}}\left(\theta,(\boldsymbol{x}_i)_{i=1:n}\right)$ w.r.t. the network parameter θ. It is given by

$$\begin{aligned}\nabla_\theta\mathcal{L}_{\mathrm{KL}}\left(\theta,(\boldsymbol{x}_i)_{i=1:n}\right) &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{J}\frac{x_{i,j}}{\pi_j(\boldsymbol{x}_i)}\nabla_\theta\pi_j(\boldsymbol{x}_i)\\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{J}\left(x_{i,j}-\pi_j(\boldsymbol{x}_i)\right)\nabla_\theta\mu_j(\boldsymbol{x}_i;\theta),\end{aligned} \tag{3.4}$$

where we have used normalization $\sum_{j=1}^{J}x_{i,j}=1$ for all $i=1,\ldots,n$. Thus, to apply the GDM we need to calculate the gradient of $\mu_j(\boldsymbol{x};\theta)$ w.r.t. the network parameter θ. State-of-the-art uses back-propagation to calculate this latter gradient, see Hastie et al. [4] or Wüthrich-Buser [10]. The GDM then provides an optimal network parameter $\widehat{\theta}$ from which we obtain the optimal probabilities

$$\boldsymbol{x}\ \mapsto\ \widehat{\pi}_j(\boldsymbol{x})=\frac{\exp\left\{\mu_j(\boldsymbol{x};\widehat{\theta})\right\}}{\sum_{j'=1}^{J}\exp\left\{\mu_{j'}(\boldsymbol{x};\widehat{\theta})\right\}}.$$

Thus, we have achieved that for all considered feature values $\boldsymbol{x}_i$ we obtain probabilities $\widehat{\pi}(\boldsymbol{x}_i)$ such that they are close together, i.e. such that the KL divergence $d_{\mathrm{KL}}\left(\boldsymbol{x}_i\|\widehat{\pi}(\boldsymbol{x}_i)\right)$ is small. The crucial idea now is to choose a neural network that has a bottleneck which means we choose $q=2$ being much smaller than the dimension J of $\mathcal{X}$, see Figure 4 (middle). This implies that $\mathcal{X}$ is projected to a two-dimensional object $(z_1(\boldsymbol{x}),z_2(\boldsymbol{x}))'$, for $q=2$, such that we can optimally rediscover $\mathcal{X}$ from these two-dimensional object (optimal w.r.t. the KL divergence). In Hinton-Salakhutdinov [5] to projection $\boldsymbol{x}\mapsto(z_1(\boldsymbol{x}),z_2(\boldsymbol{x}))'\in(-1,1)^2$ is called encoding of $\boldsymbol{x}$ and the mapping $(z_1(\boldsymbol{x}),z_2(\boldsymbol{x}))'\mapsto\pi(\boldsymbol{x})\in\mathcal{X}$ decoding of the compressed data. The explicit example is provided in Section 4, below.

3.2.2 Deep neural network approach

In the deep neural network approach we change the previous set up by adding an additional two hidden layers to the neural network, see Figure 4 (rhs) for an example. More formally, we choose $J>p>q=2$ for a bottleneck of size $q=2$. Then, we define the three hidden layers as follows. The first hidden layer is given by

$$z_l^{(1)}(\boldsymbol{x})=\tanh\left(w_{l,0}^{(1)}+\sum_{j=1}^{J}w_{l,j}^{(1)}x_j\right),\qquad\text{for } l=1,\ldots,p, \tag{3.5}$$

the second hidden layer by

$$z_l^{(2)}(\boldsymbol{x}) = \tanh\left(w_{l,0}^{(2)} + \sum_{j=1}^{p} w_{l,j}^{(2)} z_j^{(1)}(\boldsymbol{x})\right), \qquad \text{for } l = 1, \ldots, q, \tag{3.6}$$

the third hidden layer by

$$z_l^{(3)}(\boldsymbol{x}) = \tanh\left(w_{l,0}^{(3)} + \sum_{j=1}^{q} w_{l,j}^{(3)} z_j^{(2)}(\boldsymbol{x})\right), \qquad \text{for } l = 1, \ldots, p, \tag{3.7}$$

and this is then used in the regression equations

$$\mu_j(\boldsymbol{x}) = \mu_j(\boldsymbol{x}; \theta) = \beta_0^{(j)} + \sum_{l=1}^{p} \beta_l^{(j)} z_l^{(3)}(\boldsymbol{x}), \qquad \text{for } j = 1, \ldots, J. \tag{3.8}$$

Thus, we put the bottleneck layer between the two additional hidden layers each of size p. To the left of the bottleneck layer (with $q = 2$) we again have the encoding and to the right of the bottleneck layer the decoding, see Figure 4 (rhs).
This deep neural network could be calibrated brute force using the GDM. However, the GDM will often not work well because of a too high dimensionality, in fact, as described in Hinton-Salakhutdinov [5] the brute force calibration will tend to the empirical mean over all $\boldsymbol{x}_i$ (i.e. no distinction of individual differences). For this reason we calibrate this network in 3 stages,

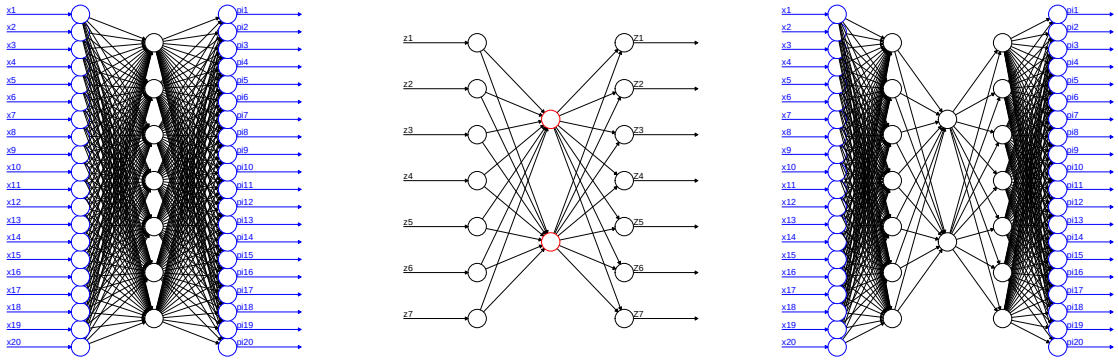

Figure 5: (lhs) shallow neural network with p ($> q$) hidden neurons for the outer calibration; (middle) shallow neural network with q hidden neurons for the inner calibration; (rhs) full deep neural network with (p, q, p) hidden neurons for the fine calibration.

see also Hinton-Salakhutdinov [5]. The idea is to decouple the neural network calibration in 3 different parts:

1. In a first part we pre-calibrate the outer weights of the deep neural network $\boldsymbol{w}^{(1)} = (w_{l,j}^{(1)})_{l=1:p,j=0:J}$ and $\boldsymbol{\beta} = (\beta_l^{(j)})_{j=1:J,l=0:p}$, given in formulas (3.5) and (3.8). This can be achieved by eliminating the middle hidden layer (with q neurons) and merging the third hidden layer with the first hidden layer. This results in fitting a shallow neural

network (3.1)-(3.2) with p ($> q$) hidden neurons in the single hidden layer. This network is illustrated in Figure 5 (lhs), and we explicitly use here that the first and third hidden layers in the deep neural network have the same number p of hidden neurons, see (3.5) and (3.7). This provides pre-calibrated weights $\boldsymbol{w}^{(1)}$ and $\boldsymbol{\beta}$.

2. Using the outer calibration from the previous step we calculate the activated neurons

$$\boldsymbol{z}^{(1)}(\boldsymbol{x}_i) = \boldsymbol{z}(\boldsymbol{x}_i) = (z_1(\boldsymbol{x}_i), \ldots, z_p(\boldsymbol{x}_i))' \in \mathbb{R}^p,$$

for our car drivers $i = 1, \ldots, n$, where $z_l(\boldsymbol{x}_i)$ is exactly obtained by (3.1) with $q = p$ and $\boldsymbol{x} = \boldsymbol{x}_i$, or by (3.5), respectively. These activated neurons $\boldsymbol{z}^{(1)}(\boldsymbol{x}_i)$ are now used to pre-calibrate the inner shallow network described by (3.6)-(3.7), this network is illustrated in Figure 5 (middle). We aim at finding weights $\boldsymbol{w}^{(2)} = (w^{(2)}_{l,j})_{l=1:q,j=0:p}$ and $\boldsymbol{w}^{(3)} = (w^{(3)}_{l,j})_{l=1:p,j=0:q}$ such that $\boldsymbol{z}^{(1)}(\boldsymbol{x}_i) \in \mathbb{R}^p$ and $\boldsymbol{z}^{(3)}(\boldsymbol{x}_i) \in \mathbb{R}^p$ are close. That is, we aim at rediscovering $\boldsymbol{z}^{(1)}(\boldsymbol{x}_i) \in \mathbb{R}^p$ as good as possible in the third hidden layer after sending the signals through the bottleneck $\boldsymbol{z}^{(2)}(\boldsymbol{x}_i) \in \mathbb{R}^q$, $q < p$. Since these hidden neurons no longer live on a unit simplex we minimize the Euclidean square distance instead of the KL divergence. Thus, determine the weights $\boldsymbol{w}^{(2)}$ and $\boldsymbol{w}^{(3)}$ such that the following square loss is minimized

$$\mathcal{L}_{\text{sq}}\left(\boldsymbol{w}^{(2)}, \boldsymbol{w}^{(3)}, (\boldsymbol{z}^{(1)}(\boldsymbol{x}_i))_{i=1:n}\right) = \sum_{i=1}^{n} \left\| \boldsymbol{z}^{(1)}(\boldsymbol{x}_i) - \boldsymbol{z}^{(3)}(\boldsymbol{x}_i) \right\|_2^2.$$

The gradient w.r.t. the network parameter $\vartheta = (\boldsymbol{w}^{(2)}, \boldsymbol{w}^{(3)})$ replacing formula (3.4) is then given by

$$\nabla_\vartheta \mathcal{L}_{\text{sq}}\left(\vartheta, (\boldsymbol{z}^{(1)}(\boldsymbol{x}_i))_{i=1:n}\right) = -2 \sum_{i=1}^{n} \sum_{j=1}^{p} \left(z_j^{(1)}(\boldsymbol{x}_i) - z_j^{(3)}(\boldsymbol{x}_i)\right) \nabla_\vartheta z_j^{(3)}(\boldsymbol{x}_i).$$

The optimal network parameter ϑ is then again determined with the GDM. This provides pre-calibrated weights $\boldsymbol{w}^{(2)}$ and $\boldsymbol{w}^{(3)}$.

3. In the previous two steps we have pre-calibrated the network parameters $\boldsymbol{w}^{(1)}$, $\boldsymbol{w}^{(2)}$, $\boldsymbol{w}^{(3)}$ and $\boldsymbol{\beta}$, illustrated by Figure 5 (lhs and middle). We use these pre-calibrated parameters as starting values in the GDM of the full deep neural network, see Figure 5 (rhs), i.e. fine-tuning of these network parameters θ is done in the last step by considering the full neural network using the pre-calibrated parameters as initial values in the back-propagation algorithm.

In the next section we provide the numerical results and compare it to the SVD.

4 Numerical results

In this section we give the explicit numerical analysis for the $n = 1753$ car drivers in speed bucket $(5, 20]$ km/h. We consider the shallow and the deep versions of the bottleneck neural

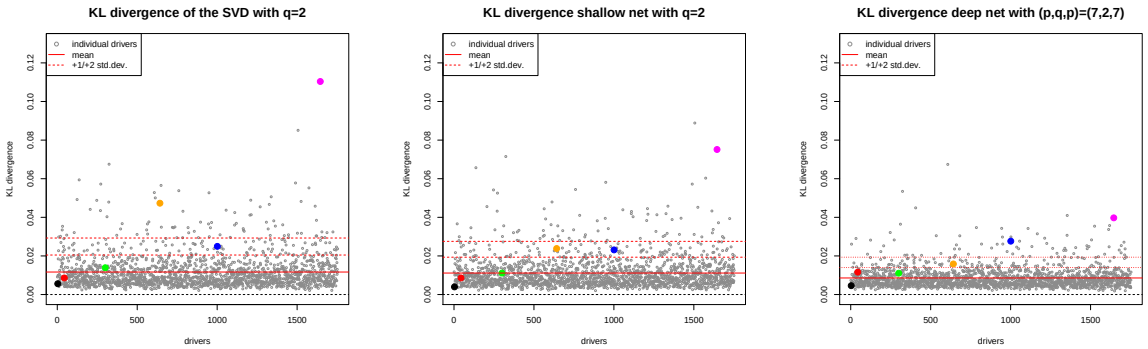

Figure 6: Individual KL divergences $d_{\mathrm{KL}}(\boldsymbol{x}_i \| \pi(\boldsymbol{x}_i))$ of all drivers $i = 1, \ldots, n$ with $i = 3, 44, 300, 642, 1001, 1645$ in black/red/green/orange/blue/magenta for (lhs) the SVD with $q = 2$, (middle) the shallow neural network with $q = 2$, and (rhs) the deep neural network with $(p, q, p) = (7, 2, 7)$.

	SVD $q = 2$	shallow $q = 2$	deep $(7, 2, 7)$	K-means
(average) KL divergence (3.3)	$11.7 \cdot 10^{-3}$	$11.1 \cdot 10^{-3}$	$8.6 \cdot 10^{-3}$	$15.0 \cdot 10^{-3}$
maximal $d_{\mathrm{KL}}(\boldsymbol{x}_i \| \pi(\boldsymbol{x}_i))$	$110.4 \cdot 10^{-3}$	$88.8 \cdot 10^{-3}$	$67.4 \cdot 10^{-3}$	$151.2 \cdot 10^{-3}$
std.dev. in KL divergence (3.3)	$8.8 \cdot 10^{-3}$	$8.2 \cdot 10^{-3}$	$5.3 \cdot 10^{-3}$	$12.1 \cdot 10^{-3}$

Table 1: KL divergence over all drivers of the SVD with $q = 2$, of the shallow neural network with $q = 2$, of the deep neural network with $(p, q, p) = (7, 2, 7)$, and of K-means with $K = 4$.

network with bottleneck $q = 2$. The results are then compared to those of the SVD with $q = 2$ principal components.

In order to compare these methods we calculate for the calibrated models the resulting multinomial logistic probabilities $\pi(\boldsymbol{x}_i) = (\pi_j(\boldsymbol{x}_i))_{j=1:J}$ from the shallow and the deep neural network calibrations, respectively. These probabilities are compared to the input values $\boldsymbol{x}_i$ using the KL divergence for $i = 1, \ldots, n$

$$d_{\mathrm{KL}}\left(\boldsymbol{x}_i \| \pi(\boldsymbol{x}_i)\right) = -\sum_{j=1}^{J} x_{i,j} \log \frac{\pi_j(\boldsymbol{x}_i)}{x_{i,j}}.$$

These numbers are compared to the corresponding numbers from the SVD. However, for the SVD we have to be careful because the resulting q-rank SVD approximation may provide a signed measure. For simplicity, we just set negative values equal to zero in the latter case. In Figure 6 we plot these KL divergences for all drivers $i = 1, \ldots, n$, the left-hand side provides the SVD result, the middle graph the shallow bottleneck neural network result, and the right-hand side the deep bottleneck neural network result. The red line gives the average KL divergences, see (3.3), and the dotted red lines indicate 1 and 2 standard deviations, respectively. Table 1 provides the corresponding numerical values.

We observe comparable results for the SVD and the shallow neural network with $q = 2$. This may be explained by the property that the shallow neural network results in a decomposition

that is close to linear. In contrast, the deep neural network provides a rather different result,

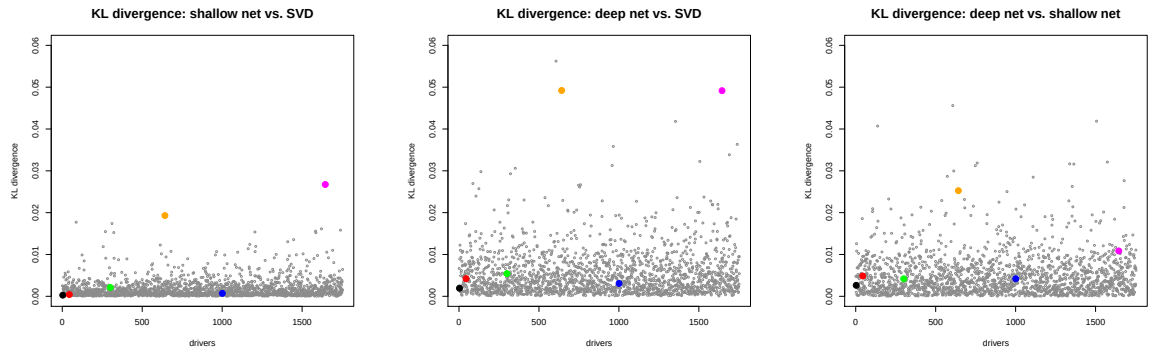

Figure 7: Individual KL divergences between (lhs) the shallow neural network and the SVD, (middle) the deep neural network and the SVD, and (rhs) the deep neural network and the shallow neural network of all drivers $i = 1, \ldots, n$ with $i = 3, 44, 300, 642, 1001, 1645$ in black/red/green/orange/blue/magenta.

see Figure 6 (rhs). The resulting KL divergences are much smaller which indicates a better rediscovery of the original features $\boldsymbol{x}_i$ of the encoding map. These differences can also be seen from Figure 7 where we plot the KL divergences between the different approximations (the scale on the y-axis is always the same). In Figures 6 and 7 we illustrate drivers $i = 3, 44, 300, 1001$ from Figure 1 and 2 in black, red, green and blue colors. We observe that for all four drivers we receive a comparably good fit, only driver $i = 1001$ is a bit difficult to be rediscovered in the deep neural network case (blue dot in Figure 6 (rhs)). In addition we show drivers $i = 642, 1645$ in orange and magenta color. From Figure 6 (lhs) we see that these two drivers are more difficult to rediscover from a linear PCA, and only the deep neural network does a decent job.
In Figure 8 we present the resulting v-a heatmaps in speed bucket $(5, 20]$ km/h of the four car drivers $i = 3, 44, 300, 1001$ (1st, 2nd, 3rd and 4th row). The first column gives the true v-a heatmap $\boldsymbol{x}_i$ (also shown in Figure 1), the 2nd column the SVD approximation $\widehat{X}_{q=2}$, the 3rd column the shallow neural network approximation $\pi(\boldsymbol{x}_i)$ and the 4th column the corresponding deep neural network approximation. Based in these color plots we see that the rediscovery of the original heatmaps is good in all three approximations, only for driver $i = 1001$ there seems to be a higher spike in the original plot in location $(v, a) = (17\text{km/h}, 0\text{m/s}^2)$. These four heatmaps correspond to the black, red, green and blue dots in Figure 6 which take comparably small values. These heatmaps should also be compared to the K-means classification results: in Figure 2 we show the average heatmaps for $k = 1, \ldots, K$ given by

$$\overline{\boldsymbol{x}}_k = \frac{\sum_{i=1}^n \boldsymbol{x}_i \mathbb{1}_{\{\text{driver } i \text{ belongs to class } k\}}}{\sum_{i=1}^n \mathbb{1}_{\{\text{driver } i \text{ belongs to class } k\}}} \in \mathcal{X}.$$

We note that also these K-means averages $\overline{\boldsymbol{x}}_1, \ldots, \overline{\boldsymbol{x}}_K$ give reasonable approximations for the four drivers $i = 3, 44, 300, 1001$.
In Figure 9 we again give the KL divergences of Figure 6 (lhs) and (rhs) of the SVD and the

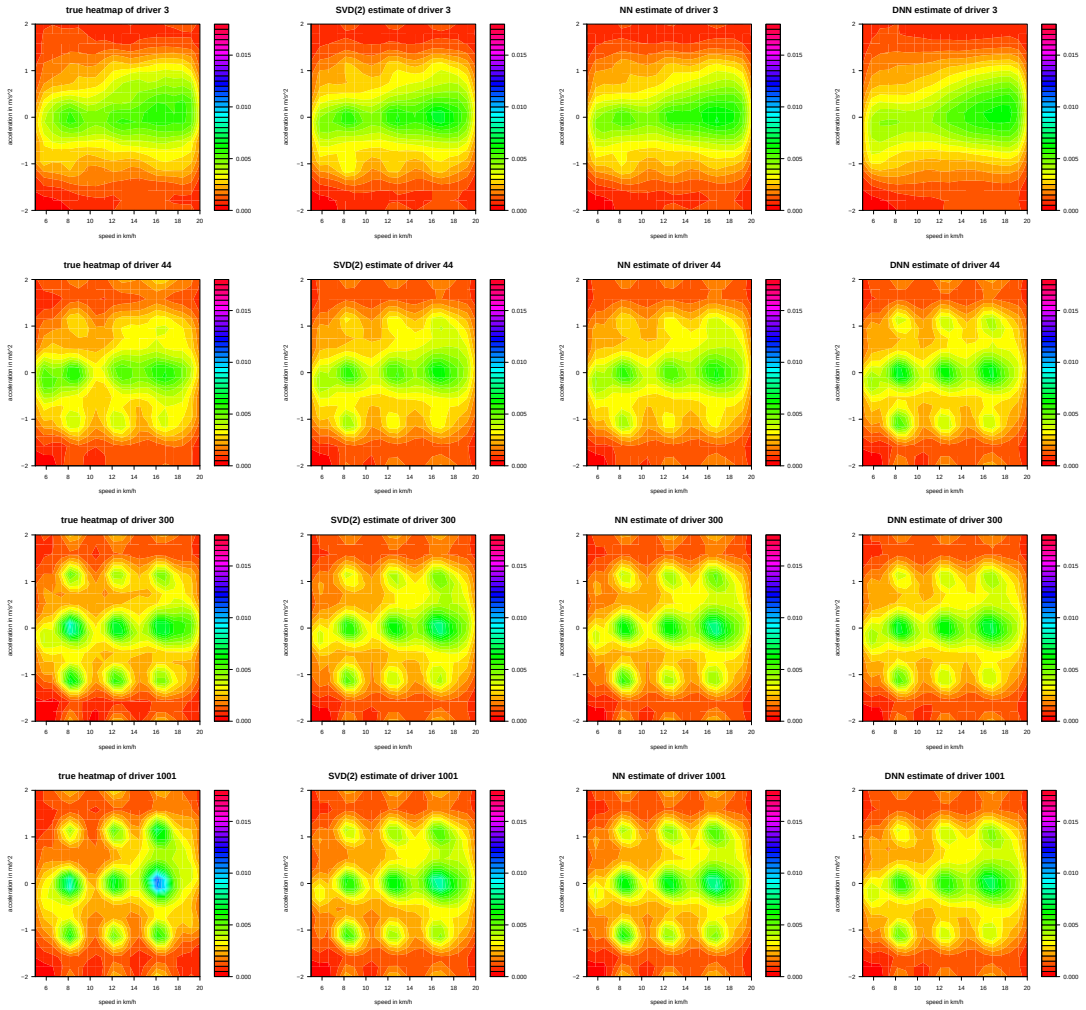

Figure 8: v-a heatmaps in speed bucket $(5, 20]$ km/h of car drivers $i = 3, 44, 300, 1001$ (1st, 2nd, 3rd and 4th row): the 1st column is the true heatmap (see Figure 1), the 2nd column the SVD approximation, the 3rd column the shallow neural network approximation and the 4th column the deep neural network approximation.

deep neural network approximations. However, we change the scale on the y-axis in Figure 9 so that we can compare them to the KL divergences coming from the K-means approximations $\overline{\boldsymbol{x}}_1, \ldots, \overline{\boldsymbol{x}}_4$ in the $K = 4$ categorical classes. The latter is presented in Figure 9 (middle). Besides the low predictive power of the K-means approximation we observe that there are some outliers that substantially deviate from the class means $\overline{\boldsymbol{x}}_k$. In particular, car drivers $i = 642, 1645$ in orange and magenta color in Figure 9 cause difficulties in approximation. Car driver $i = 614$ belongs to class $k = 2$ and car driver $i = 1645$ to class $k = 4$. These two car drivers are still difficult to be captured by the SVD, see Figure 9 (lhs), but the deep neural network approach seems to be able to deal reasonably well with these two drivers. In Figure 10 we provide the v-a heatmaps of these two car drivers $i = 642, 1645$, together with their approximations. We note that the deep neural network is clearly better for these two car drivers (though not perfect).

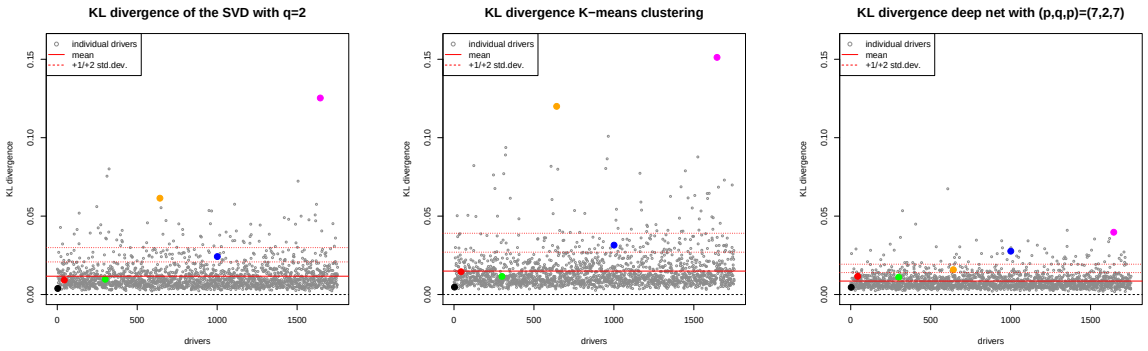

Figure 9: Individual KL divergences of all drivers $i = 1, \ldots, n$ with $i = 3, 44, 300, 642, 1001, 1645$ in black/red/green/orange/blue/magenta against (lhs) the SVD with $q = 2$, (middle) the K-means averages $\overline{\boldsymbol{x}}_k$, and (rhs) the deep neural network with $(p, q, p) = (7, 2, 7)$.

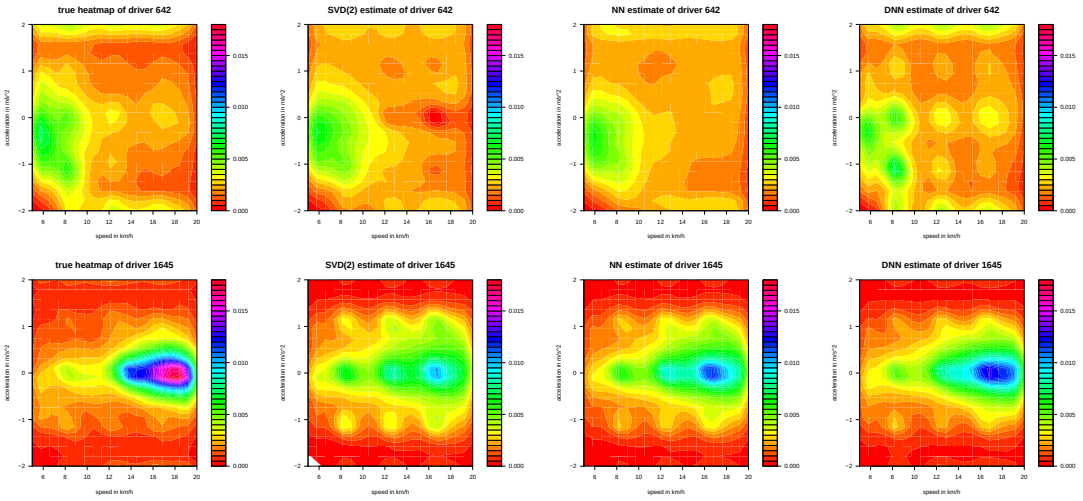

Figure 10: v-a heatmaps in speed bucket $(5, 20]$ km/h of car drivers $i = 642, 1645$ (1st and 2nd row): the 1st column is the true heatmap (see Figure 1), the 2nd column the SVD approximation, the 3rd column the shallow neural network approximation and the 4th column the deep neural network approximation.

Finally, we come to the main result presented in Figure 11. This figure gives the two-dimensional representations of all $n = 1753$ car drivers' v-a-heatmaps. The left column shows the first two principle components of the SVD, the colored dots on the 1st row giving the same six car drivers $i = 3, 44, 300, 642, 1001, 1645$ as in Figure 6. We observe that car drivers $i = 642, 1645$ (orange/magenta) are outliers, as expected, and the other car drivers do not stand out. The 2nd row in Figure 11 shows the same two-dimensional representation, but the dots are colored according to their categorical classes from the K-means classification given in Figure 2. The middle and the right columns give the same results for the shallow and the deep bottleneck neural networks. The two-dimensional locations of the car drivers i are given by the corresponding

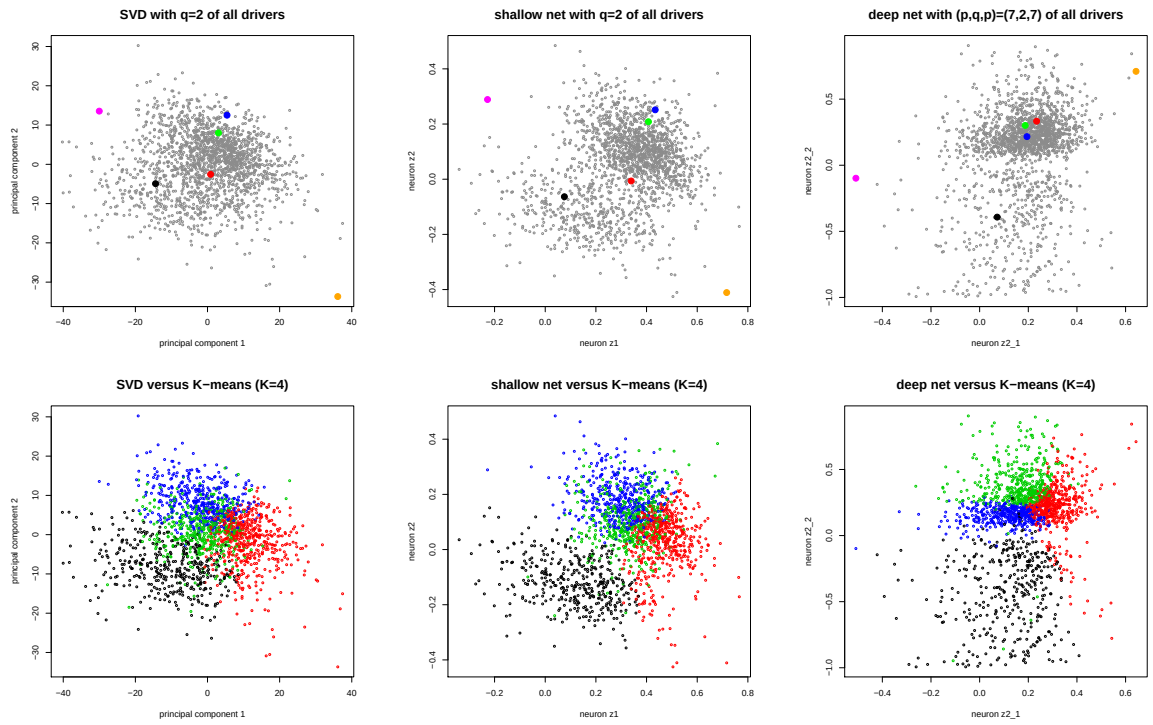

Figure 11: Two-dimensional representations of the $n = 1753$ car drivers' v-a-heatmaps: (lhs) SVD showing the first two principle components, (middle) shallow neural network showing the network activation $(z_1(\boldsymbol{x}_i), z_2(\boldsymbol{x}_i))'$, and (rhs) deep neural network showing the network activation $(z_1^{(2)}(\boldsymbol{x}_i), z_2^{(2)}(\boldsymbol{x}_i))'$; the 1st row highlights the car drivers $i = 3, 44, 300, 642, 1001, 1645$ as in Figure 6, the 2nd row colors the car drivers according to their K-means classes 1 (black), 2 (red), 3 (green) and 4 (blue).

activations $(z_1(\boldsymbol{x}_i), z_2(\boldsymbol{x}_i))'$ and $(z_1^{(2)}(\boldsymbol{x}_i), z_2^{(2)}(\boldsymbol{x}_i))' \in (-1, 1)^2$ at the bottlenecks. Remarkable is that the deep neural network activation give a clear separation of the car drivers which is very similar to the K-means result (for $K = 4$), notice that the colored boundaries in Figure 11 (2nd row, rhs) are rather sharp.

5 Conclusions

We conclude that the SVD and bottleneck neural networks achieve to provide low-dimensional continuous representations of complicated objects such as v-a heatmaps. This is interesting in insurance for several reasons. These continuous representations can now be used in regression models as continuous covariates. This has the advantage that, typically, the number of necessary regression parameters is lower compared to categorical covariates. This leads to less over-parametrization.

Another nice consequence is that we can now simulate v-a heatmaps from two-dimensional distributions. Namely, we can simulate points $\boldsymbol{z} \in (-1, 1)^2$ and then use the decoder that maps

these two-dimensional points at the bottleneck to the corresponding probabilities $\pi \in \mathcal{X}$ by propagating these signals to the output layer of the neural networks (and of course analogously for the SVD).

There remains to prove that these v-a heatmaps have predictive power to forecast potential car driving claims. Based on this data this is not possible because we do not have insurance claims information. In our work we have only explored unsupervised learning methods that provide pattern recognition which is useful in designing (low-dimensional) covariates.

A KL divergence, revisited

In this appendix we briefly revisit the KL divergence. Denote by $\mathcal{X} \subset \mathbb{R}^J$ the J-unit simplex. Consider k independent and identically distributed trials among J classes providing a multinomial distribution $\pi \in \mathcal{X}$ given by the discrete probability weights

$$p(k_1, \ldots, k_J) \;=\; \binom{k}{k_1, \ldots, k_J} \prod_{j=1}^{J} \pi_j^{k_j} \cdot \mathbb{1}_{\{\sum_{j=1}^{J} k_j = k\}},$$

for $k_j \in \mathbb{N}_0$, $j = 1, \ldots, J$. The deviance statistics of an observation $(k_1, \ldots, k_J)$ of that multinomial distribution is given by

$$D\left((k_1, \ldots, k_J), \pi\right) = 2\left[\sum_{j=1}^{J} k_j \log\left(\frac{k_j}{k}\right) - \sum_{j=1}^{J} k_j \log \pi_j\right] = 2k \sum_{j=1}^{J} \frac{k_j}{k}\left(\log\left(\frac{k_j}{k}\right) - \log \pi_j\right).$$

In Section 3 we have defined the empirical distributions on the J-unit simplex by setting $x_j = k_j/k$ which, of course, provides $\boldsymbol{x} = (x_1, \ldots, x_J)' \in \mathcal{X}$. Doing this transformation we receive

$$D\left((k_1, \ldots, k_J), \pi\right) = -2k \sum_{j=1}^{J} x_j \log \frac{\pi_j}{x_j} = 2k \; d_{\mathrm{KL}}\left(\boldsymbol{x} \| \pi\right).$$

Thus, by minimizing the KL divergence in (3.3), we minimize the corresponding deviance statistics, which provides the maximum likelihood estimator of the network parameter θ under independent multinomial models (having J classes) for drivers $i = 1, \ldots, n$. This additionally assumes that all drivers have identical weights k. If the latter is not appropriate we may replace the average KL divergence in (3.3) by a weighted counterpart

$$\mathcal{L}_{\mathrm{KL}}^{w}\left(\theta, (\boldsymbol{x}_i)_{i=1:n}\right) = \sum_{i=1}^{n} w_i \; d_{\mathrm{KL}}\left(\boldsymbol{x}_i \| \pi(\boldsymbol{x}_i)\right),$$

with weights $w_i \geq 0$ satisfying $\sum_{i=1}^{n} w_i = 1$.

References

[1] Ayuso, M., Guillen, M., Pérez-Marín, A.M. (2016). Telematics and gender discrimination: some usage-based evidence on whether men's risk of accidents differs from women's. *Risk* **4/2**, article 10.

[2] Kramer, M.A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal* **37/2**, 233-243.

[3] Hainaut, D. (2017). A neural network analyzer for mortality forecast. *Preprint.*

[4] Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* 2nd edition. Springer Series in Statistics.

[5] Hinton, G.E., Salakhutdinov, R.R. (2006). Reducing the dimensionality of data with neural networks. *Science* **313**, 504-507.

[6] Verbelen, R., Antonio, K., Claeskens, G. (2016). Unraveling the predictive power of telematics data in car insurance pricing. *SSRN Manuscript* ID 2872112.

[7] Weidner, W., Transchel, F.W.G., Weidner, R. (2016). Classification of scale-sensitive telematic observables for riskindividual pricing. *European Actuarial Journal* **6/1**, 3-24.

[8] Weidner, W., Transchel, F.W.G., Weidner, R. (2016). Telematic driving profile classification in car insurance pricing. *Annals of Actuarial Science* **11/2**, 213-236.

[9] Wüthrich, (2017). Covariate selection from telematics car driving data. *European Actuarial Journal* **7/1**, 89-108.

[10] Wüthrich, M.V., Buser, C. (2016). Data analytics for non-life insurance pricing. *SSRN Manuscript* ID 2870308. Version October 25, 2017.